A Massively Parallel Computation Strategy for FDTD: Time and Space Parallelism Applied to Electromagnetic Problems

Amir **Fijany***, Michael A. Jensent, **Yahya** Rahmat-Samiit, and Jacob Barhen*
* Jet Propulsion Laboratory, California Institute of Technology
t Department of Electrical Engineering, University of California, Los Angeles

Abstract

In this paper, we present a novel strategy for incorporating massive parallelism into the solution of Maxwell's equations using finite-difference time-domain methods. In a departure from previous techniques wherein *spatial* parallelism is used, our approach exploits *massive temporal* parallelism by computing all of the time steps in parallel. Furthermore, in contrast to other methods which appear to concentrate on explicit schemes, our strategy uses the implicit Crank-Nicolson technique which provides superior numerical properties. We show that the use of temporal parallelism results in algorithms which offer a massive degree of coarse grain parallelism with minimum communication and synchronization requirements. Due to these features, the time-parallel algorithms are particularly suitable for implementation on emerging massively parallel MIMD architectures. The methodology is applied to a circular cylindrical configuration – which serves as a testbed problem for the approach – to demonstrate the massive parallelism that can be exploited. We also discuss the generalization of the methodology for more complex problems.

Submitted to the *IEEE Transactions on Antennas and Propagation*. A short version will be presented at the IEEE International Symposium on Antennas and Propagation, Seattle, WA, June 1994.

1 Introduction

The application of finite-difference and finite-volume time-domain (FDTD and FVTD) methods to the solution of Maxwell's equations has been encouraged by increased efforts to model elect rically large and complex radiators and scatterers. However, because of the intensive computation and storage associated with these techniques, their practical implementation for very large problems presents some challenges. In Miller's survey of computational electromagnetic (CEM) techniques [1], it is suggested that a key solution to the computational power and storage requirement bottlenecks is the exploitation of a massive degree of parallelism by implementing CEM FDTD algorithms on parallel architectures. However, in order to fully exploit the computing power oRered by available parallel platforms, existing algorithms must be reexamined, with a focus on their efficiency for parallel implementation. Eventually, new algorithms may have to be developed that, from the onset, take a greater advantage of the available massive parallelism.

An emerging class of massively parallel multiple instruction multiple data (MIMD) supercomputers, such as Intel's Delta and Paragon and CRAY's T3D, appears to set the current and future trend in massively parallel computing technology. The very large number of vector processors employed in these supercomputers couples the high level, coarse grain parallelism of the MIMD architecture with a lower level vector processing capability to provide an impressive computational throughput. The main limitation of these architectures lies in their rather limited communication structure (mesh for Delta and T3D). This feature makes them most suitable for algorithms which possess a high degree of coarse grain parallelism, require limited communication and synchronization, and involve basic operations (or algorithmic processes) that can be efficiently vectorized. The algorithms presented in this paper have been developed based upon this model of computation and, as such, are highly efficient for implementation on this class of massively parallel MIMD supercomputers.

1.1 Background of FDTD CEM

FDTD and FVTD techniques are generally derived by approximating derivatives in Maxwell's time-domain coupled partial differential equations (PDEs) using finite differences. The resulting algebraic equations are then used to track the evolution of the fields in a region of space using time-stepping procedures. A number of such schemes [2]-[6] have been developed and used in CEM, perhaps the most popular of which was originally proposed by Yee [2] and later developed by others [3, 4]. The majority of these techniques are explicit and require a matrix-vector multiplication, wherein the matrices are highly sparse, at each time step. This fact suggests that they offer a high degree of spatial parallelism - the exploitation of parallelism at the computation of each time step - and therefore a considerable amount of work has been devoted to the space-parallel implementation of several FDTD methods. However, such a parallelism is rather fine grain and, due to communication and synchronization requirements, it can not be efficiently exploited by massively parallel MIMD architectures. In fact, practical implementations of Yet's algorithm on computers such as the Hypercube [7, 8] and Transputer [9] clearly show that computational speedup is limited since only a few processors can be efficiently employed. In contrast, these methods *are* highly suitable for vector processing, and as such may be efficiently computed using a high degree of vectorization but a limited degree of parallelism. The results reported in [6] and [1 O] appear to support the optimality of such a strategy.

With the exception of [11] which illustrates the application of the Crank-Nicolson (CN) method to the scalar wave equation, it seems that less attention has been focussed on the application of implicit [12] methods to CEM problems. These methods offer the advantage of unconditional stability, which often reduces the number of time steps required. However, because implicit techniques require a linear system solution at each time step, it has generally been assumed that they are inefficient for parallel computation (see Section 2 for a more detailed discussion).

1.2 Time-Parallel Algorithm

In this paper, we propose a novel computational strategy which uses time-parallelism - the exploitation of parallelism in the computation of all the time steps – in the CN solution of the scalar wave equation. The resulting time-parallel algorithm offers a high degree of coarse grain parallelism with minimum communication and synchronization requirements, making it highly efficient for implementation on massively parallel MIMD architectures. The application of our time-parallel computing approach to determine the electromagnetic behavior of fields near circular cylindrical geometries clearly reveals the massive temporal parallelism which can be efficiently exploited in the computation. It is further shown that, with the availability of a larger number of processors, spatial parallelism can be also exploited in the computation, resulting in a time- and space-parallel algorithm which remains highly coarse grain with a simple communication structure.

Although our approach is not yet as general as more established methods, we believe that our work paves the way for a new direction in massively parallel CEM. In this sense, this paper mainly presents the basic idea underlying the time-parallel computing approach along with a discussion on its efficient applicability by considering a representative problem. However, the results of this paper provide a framework for further research work on the application of this strategy to a wider class of CEM problems.

This paper is organized as follows. Section 2 reviews some fundamental ideas relating to the parallel solution of time-dependent PDEs and introduces the underlying concepts behind time-parallel computation. In Section 3, the algorithm is applied to the circular cylinder. The performance of the time-parallel algorithm with respect to the best sequential explicit and implicit methods for the same cylinder problem is analyzed in Section 4. Generalization of the time-parallel computing approach is discussed in Section 5. Finally, Section 6 provides some concluding remarks.

2 Time-Parallel Algorithm for FDTD CEM

2.1 Parallel Solution of Time-Dependent PDEs

The computational basis for the implementation of the time-parallel algorithm is the time-dependent form of Maxwell's coupled PDEs. For homogeneous, isotropic materials, these equations can be written as two decoupled second-order wave equations which assume the form

$$\begin{aligned} \frac{\partial^2 \vec{E}}{\partial t^2} &= c^2 \nabla^2 \vec{E} \\ \frac{\partial^2 \vec{H}}{\partial t^2} &= c^2 \nabla^2 \vec{H} \end{aligned} \tag{2.1}$$

where ∇^2 is the Laplace operator, c is the speed of light, and $\vec{E}$ and $\vec{H}$ represent the electric and magnetic field intensities, respectively. Depending on the problem geometry, a set of scalar hyperbolic equations can be selected from Eq. (2.1). These equations can be expressed in a general form as

$$\frac{\partial^2 \psi}{\partial t^2} = \nabla^2 \psi + g(\vec{r}, t) \qquad \vec{r} \in \Omega,\ 0 < t < T \tag{2.2}$$

where Ω is a bounded domain with boundary Ω', $g(\vec{r}, t)$ indicates a time and space dependent source term, and where the initial and boundary conditions are specified. Using finite difference approximations for the derivatives results in a general discretized form of Eq. (2.2) which may be written as

$$A\psi^{(m+1)} = B\psi^{(m)} + C\psi^{(m-1)} + f^{(m+1)} \qquad 1 \le m \le M - 1 \tag{2.3}$$

where $\psi^{(m)}$ is the approximate solution at the mth time step, At is the time step size, and $M = T/At$. The term $f^{(m+1)}$ results from the discretization of $g(\vec{r}, t)$ in time and space, and $\psi^{(0)}$ and $\psi^{(1)}$ are the given initial conditions. In addition, we will use the symbol N to represent the size of the spatial grid. The specific structure of the matrices *A*, *B*, and C and the computation of $f^{(m+1)}$ depend upon the solution method as well as the time and space discretization strategies employed. Eq. (2.3) provides a basis for discussion of the exploitation of time-parallelism in FDTD approximations to the hyperbolic wave equation.

Although most of the current research on developing parallel techniques for solution of time-dependent PDEs appears to concentrate on parabolic equations, many of the techniques developed can be extended to the solution of hyperbolic equations. Much of this work is motivated by three widely acknowledged observations (see for example [13]- [1 5]) regarding the efficiency of time-stepping methods for parallel computation:

1. Explicit methods, while limited in their range of stability, are highly efficient for parallel and/or vector processing since the computation at each time step mainly involves a matrix-vector multiplication.

2. Implicit and CN methods, despite their superior numerical properties, are not efficient for parallel and/or vector processing since at each time step a linear system solution is required.

3. The implementation of time-stepping methods is generally assumed to be strictly sequential in time, implying that the solution for time step $m + 1$ can not be obtained without first computing the solution for step m.

The first observation has motivated the development of new explicit methods which offer improved numerical properties while preserving the efficiency for parallel/vector computation [16, 17]. The second observation has resulted in new techniques which improve the efficiency of implicit methods for parallel computation while preserving their numerical properties [14, 18]. However, these algorithms can be classified as space-parallel since they attempt to parallelize the computation at each time-step while the overall computation remains strictly sequential in time. Finally, the third observation has motivated the investigation of new iterative techniques to increase parallelism in time [19] -[24]. However, the resulting algorithms achieve a rather limited temporal parallelism. In fact, Womble [21] supports the assessment of [25] wherein simultaneous solution for all time steps is not considered feasible.

However, we have recently shown that, for a wide class of time-dependent PDEs, the computation of time-stepping methods can be fully parallelized in time, leading to a massive degree of temporal parallelism in the computation [26] -[28]. The practical application of

the time-parallel algorithm to a simple problem (a two dimensional heat equation) on Intel's Touchstone Delta has shown that linear and even super-linear speedup can be achieved by using a very large number of processors (of the order of 102) [29]. In the following we discuss the implementation of the time-parallel algorithm to the CN solution of Eq. (2.2).

2.2 Temporal Parallelism

To motivate the idea of time-parallelism, notice that Eq. (2.3) simply represents a Second-Order Inhomogeneous Linear Recursion (SOILR) which can be cast into a first-order one. The solution of such recurrences can be fully parallelized and computed in $O(\log M)$ by using the Recursive Doubling Algorithm (RDA) [30] or Cyclic Reduction Algorithm (CRA) [31]. Unfortunately, such an approach is not *computationally practical* since both RDA and CRA compute powers and products of the matrices in Eq. (2,3), resulting in increasingly dense and ultimately full matrices. For a three dimensional problem, this will involve the multiplication of dense $N^3 \times N^3$ matrices with a computation complexity of $O(N^9)$! Even if the computation is fully parallelized in time, the cost of one such matrix-matrix multiplication alone would be much greater than the cost of any serial algorithm. Nevertheless, this observation clearly indicates that, insofar as the data dependency in the computation is concerned, the time-stepping procedures can be fully parallelized in time.

Motivated by this observation, we have developed a technique which allows the *efficient* time-parallelization of time-stepping methods, leading to a highly practical approach for massively parallel computation. To describe this technique, consider the CN method for solution of Eq. (2.2). In this case, Eq. (2.3) is written as

$$(I - \alpha M)\psi^{(m+1)} = 2I\psi^{(m)} - (I - \alpha M)\psi^{(m-1)} + f^{(m+1)} \qquad 1 < m < M - 1 \qquad (2.4)$$

where I is the unit matrix, α is a constant, and M is the matrix arising from the discretization of the Laplace operator. The source vector $f^{(m+1)} = \zeta[g^{(m+1)} + g^{(m-1)}]$, where ζ is a constant, represents the discretized source term in Eq. (2.2). Now, let the Eigenvalue/Eigenvector (EE) decomposition of the nonsingular matrix M be given by

$$M = \Theta \Lambda \Theta^{-1} \qquad (2.5)$$

where Θ is the set of eigenvectors and A is a diagonal matrix representing the set of eigenvalues of M. Substituting Eq. (2.5) into Eq. (2.4) and taking Θ and Θ^{-1} outside the parentheses gives the expression

$$\Theta(I - \alpha\Lambda)\Theta^{-1}\psi^{(m+1)} = 2I\psi^{(m)} - \Theta(I - \alpha\Lambda)\Theta^{-1}\psi^{(m-1)} + f^{(m+1)}. \quad (2.6)$$

Multiplying both sides of Eq. (2.6) by the nonsingular matrix Θ^{-1}, we obtain

$$(I - \alpha\Lambda)\Theta^{-1}\psi^{(m+1)} = 2I\Theta^{-1}\psi^{(m)} - (I - \alpha\Lambda)\Theta^{-1}\psi^{(m-1)} + \Theta^{-1}f^{(m+1)}. \quad (2.7)$$

Defining a diagonal matrix S = $(I - \alpha\Lambda)^{-1}$ and the vectors $\tilde{\psi}^{(m)} = \Theta^{-1}\psi^{(m)}$ and $\tilde{f}^{(m)} = S\Theta^{-1}f^{(m)}$ allows us to write Eq. (2.7) as

$$\tilde{\psi}^{(m+1)} = 2S\tilde{\psi}^{(m)} - \tilde{\psi}^{(m-1)} + \tilde{f}^{(m+1)}. \quad (2.8)$$

In contrast to Eq. (2.4), Eq. (2.8) is diagonalized and can therefore be efficiently solved in parallel using RDA or CRA. For two and three dimensional problems, Eq. (2.8) can be computed in $O(N^2 \log M)$ and $O(N^3 \log M)$ by using $O(M)$ processors.

2.3 Structure of the Time-Parallel Algorithm

To facilitate the discussion of the time-parallel algorithm, we subdivide it into four steps as illustrated in Figure 1. Step 1 involves determining the EE decomposition of the matrix M and forming the matrix S. The source vectors $f^{(m)}$ are computed and multiplied by $S\Theta^{-1}$ in Step 2 to obtain the vectors $\tilde{f}^{(m)}$. The parallel solution of the SOILR in Eq. (2.8) is accomplished in Step 3. Finally, the solution vectors $\psi^{(m)}$ are obtained in Step 4 by performing the matrix-vector multiplication $\Theta\tilde{\psi}^{(m)}$.

Several key issues relating to these steps must be addressed before applying the approach to an example problem. To begin, since the computation in Step 1 typically only generates Θ and A, the multiplication by Θ^{-1} in Step 2 involves a linear system solution at each time step. However, if Θ^{-1} can be obtained explicitly, then Step 2 becomes a sequence of matrix-vector multiplications. Determination of Θ^{-1} is simply performed if M is symmetric since $\Theta^{-1} = \Theta^T$ (where T indicates transpose). If M is nonsyrnmetric, the fact that $\mathbf{M}^T = (\Theta^{-1})^T\Lambda\Theta^T$

suggests that the matrix Θ^{-1} can be obtained by computing the EE decomposition of MT in Step 1. This can be performed in parallel with the EE decomposition of M without increasing the overall computational complexity, Because the computation in Step 1 is space dependent, it needs to be performed only once for a given problem geometry.

The transformations in both Step 2 and Step 4 are completely decoupled and can be performed in parallel with no communication among processors. Assuming $O(M)$ processors are available, each of these steps requires a complexity of 0(1) in time and $O(\kappa)$ in space – where κ denotes the cost of a matrix-vector multiplication. The communication required in the computation of Step 3 has a rather simple structure and can be efficiently implemented on MIMD parallel architectures since it involves the exchange of large vectors among processors. For practical values of *M* and N and for most cases, the computational cost of Step 3 is much less than that of Steps 2 and 4 (see for example Section 3). This implies that the overall complexity of the time-parallel algorithm is dominated by the computations of Steps 2 and 4 which can be fully parallelized in time. This also illustrates the highly decoupled structure, coarse grain size, and vector nature of the scheme.

These features of the time-parallel algorithm have motivated us to identify the class of problems to which the scheme may be *efficiently* applied. Two key requirements for the efficient application of the method are:

1. an efficient scheme for determination of the eigenpairs of M, and

2. an efficient scheme for the matrix-vector multiplications in Steps 2 and 4.

The second issue is particularly important since multiplication of a dense matrix by a vector leads to a computational cost of $O(N^4)$ and $O(N^6)$ for two and three dimensional problems respectively. This fact, coupled with the first issue, motivates us to exploit the sparse structure of M and use factored forms of Θ and Θ^{-1} to increase the efficiency of these steps.

Taking into account these two issues, our analysis indicates that, for a class of problems, the time-parallel algorithm can be *readily* applied with an optimal efficiency. In these cases, the EE decomposition of M can be computed efficiently with a high degree of parallelism.

Furthermore, the matrix of eigenvectors can be obtained as a product of highly sparse matrices which reduces the complexity of the matrix-vector multiplication. In order to further clarify the implications of these two issues, we first consider the application of the time-parallel algorithm to a specific problem. Generalization of the time-parallel algorithm is discussed in Section 5.

2.4 Absorbing Boundary Condition.

For many problems, particularly enclosed problems involving resonant cavities or waveguiding structures, the time-parallel algorithm may be readily applied since typically either a Dirichlet or Neumann condition is enforced on the outer spatial domain boundary. However for open problems encountered in radiation and scattering, the issue of incorporating an absorbing boundary condition (ABC) [32] at the outer domain boundary must be addressed. Without proper termination of the computational grid, outward traveling waves from the structure will be artificially reflected by the grid truncation. For the time-parallel algorithm, proper choice of an ABC is of critical importance. This arises from the fact that the derivation of the scheme depends upon the fundamental assumption that all matrices in Eqs. (2.3) and (2.4) are simultaneously diagonalizable. While this assumption holds for conventional boundary conditions (i.e. Dirichlet, Neumann, Mixed, and Periodic), it is not necessarily valid for all existing forms of ABCs.

In light of this fact, an ABC which is consistent with the structure of the time-parallel algorithm involves solving the problem once for a Dirichlet and once for a Neumann boundary condition at the outer domain boundary [33]. The results of the two computations are then averaged to give the desired solution. This scheme not only maintains the diagonalizability of Eq. (2.4) but also is highly suitable for parallel computation since the two solutions can be performed in parallel. Recently, it has been reported that this technique provides satisfactory results when applied to FDTD solution of Maxwell's equations [34]. However, our investigation has shown that while this ABC is effective for short time-stepping durations, it suffers from inaccuracies when the run-time is long enough for multiple reflections to occur.

For this reason, additional research is underway to identify improved techniques for including non-reflecting boundary conditions within the framework of the time-parallel algorithm.

3 Algorithm Implementation for a Circular Cylinder

Although the high-level structure of the time-parallel algorithm, as outlined in Section 2.3, is the same for various applications, some of the details of the implementation may change from problem to problem. In this section, we illustrate the method by applying it to the CN solution of Maxwell's equations for circular cylindrical geometries. The concepts illustrated here can be used to compute the behavior of coaxial cavity configurations or the scattering from circular cylinders depending on the choice of boundary conditions used. The key area of emphasis in this demonstration involves determining the EE decomposition of M and diagonalizing the resulting CN matrix equation.

3.1 Laplace Operator in Polar Coordinates

Let ρ_0 and ρ_1 represent the radii of the cylinder and the edge of the finite computational domain, as shown in Figure 2. In this case, the domain Ω of Eq. (2.2) is an annulus between $p_.$ and $p_{l.}$ Also let ψ represent E_z, the 2-polarized electric field surrounding the circular cylinder. The Laplace operator in polar coordinates is given by

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial\rho^2} + \frac{1}{\rho}\frac{\partial\psi}{\partial\rho} + \frac{1}{\rho^2}\frac{\partial^2\psi}{\partial\phi^2}. \tag{3.1}$$

The mesh points in the ρ-ϕ plane are defined by the points of intersection of the circles $p = i\Delta\rho$, $q - 1 < i \leq p + 1$, and the straight lines $\phi = j\Delta\phi$, $1 \leq j \leq N$, where $\Delta\phi = 2\pi/N$. For simplicity, it is assumed that $\rho_0 = (q - 1)\Delta\rho$, $p_l = (p+1)\Delta\rho$ and $K = p - q + 1$. Using this notation along with the convention that $\psi(i\Delta\rho, j\Delta\phi, m\Delta t) = \psi_{i,j}^{(m)}$, the five-point finite difference approximation of Eq. (3. 1) becomes

$$\begin{aligned}\nabla^2\psi(i,j) = \frac{1}{(\Delta\rho)^2}\Bigg\{&\left(1+\frac{1}{2i}\right)\psi_{i+1,j} + \left(1-\frac{1}{2i}\right)\psi_{i-1,j} - \\ &2\left[1+\frac{1}{(i\Delta\phi)^2}\right]\psi_{i,j} + \frac{1}{(i\Delta\phi)^2}(\psi_{i,j+1}+\psi_{i,j-1})\Bigg\}.\end{aligned} \tag{3.2}$$

If the field scattered from the cylinder is to be computed, the boundary condition $\psi_{q-1,j}^{(m)} = -E_z^{inc}(\rho_0, j\Delta\phi, m\Delta t)$ must be satisfied on the inner domain boundary, where E_z^{inc} represents the incident plane wave (assuming TM^z incidence). In the CN matrix equation, this is accomplished by using the source vector $f^{(m)}$.

The vectors $f^{(m)}$ and $\psi^{(m)} \in \mathbf{R}^{KN}$, which are based on ordering the vector elements first in the direction of p and then in the direction of ϕ, are defined as

$$\begin{aligned} f^{(m)} &= \text{col}\{f_i^{(m)}\} \text{ and } \psi^{(m)} = \text{col}\{\psi_i^{(m)}\}, \; q \le i \le p, \\ f_i^{(m)} &= \text{col}\{f_{ij}^{(m)}\} \text{ and } \psi_i^{(m)} = \text{col}\{\psi_{ij}^{(m)}\}, \; 1 < j \le N. \end{aligned} \tag{3.3}$$

An alternative representation of these vectors which will be used is based upon ordering first in the direction of ϕ and then in the direction of ρ, leading to the forms

$$\begin{aligned} f^{(m)} &= \text{col}\{f_j^{(m)}\} \text{ and } \psi^{(m)} = \text{col}\{\psi_j^{(m)}\}, \; 1 \le j \le N, \\ f_j^{(m)} &= \text{col}\{f_{ji}^{(m)}\} \text{ and } \psi_j^{(m)} = \text{col}\{\psi_{ji}^{(m)}\}, \; q \le i \le p. \end{aligned} \tag{3.4}$$

3.2 EE Decomposition: Dirichlet Boundary Condition

Using Eq. (3.2) with the Dirichlet boundary condition at the outer domain boundary, we obtain a block tridiagonal matrix M given by

$$\text{M} = \text{Tridiag}\,[(1 - 1/2i)I \;\; B_i \;\; (1 + 1/2i)I] \in \mathbf{R}^{KN \times KN}. \tag{3.5}$$

The submatrix $B_i \in \mathbf{R}^{N \times N}$ is given by

$$B_i = \beta_i B - 2I \tag{3.6}$$

where $\beta_i = (1/i\Delta\phi)^2$ and B is a N x N matrix given by

$$B = \begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ \vdots & \vdots & \vdots & & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}. \tag{3.7}$$

This specific structure of B results from the periodicity in ϕ.

The following theorem is used in deriving the EE decomposition of M.

Theorem 1 *The* ***EE*** *decomposition of the* ***matrix*** *B is given by*

$$B = FDF^{-1} \tag{3.8}$$

where F is the matrix of ***eigenvectors*** *and D =* $Diag\{d_j\}$, $1 \leq j \leq N$ *is the diagonal matrix* ***of eigenvalues of*** *B, with* $d_j = -4 \sin^2[(j-1)\pi/N]$ *[35, p. 253].*

The matrices *F* and *F-l* are the direct and inverse one dimensional Discrete Fourier Transform (DFT) operators [35]. Thus, multiplication of any vector by these operators corresponds to performing a DFT operation which, by using fast techniques, can be performed in O(N log N). From the definition in Eq. (3.6), it follows that B_i and *B* share the same set of eigenvectors but have a different set of eigenvalues. In light of this, the EE decomposition of B_i is given by

$$B_i = F\lambda_i F^{-1} \tag{3.9}$$

where $\lambda_i = \text{Diag}\{\lambda_{i,j}\} = \beta_i D - 2I$, $1 \leq j \leq N$, and

$$\lambda_{i,j} = \beta_i d_j - 2 = -4\beta_i \sin^2[(j-1)\pi/N] - 2. \tag{3.10}$$

Let $\mathcal{F} = \text{Diag}\{F, F, \ldots, F\}$ and $\mathcal{F}^{-1} = \text{Diag}\{F^{-1}, F^{-1}, \ldots, F^{-1}\} \in \mathbf{R}^{KN \times KN}$. Also, consider a permutation matrix $\mathcal{P} \in \mathrm{R}^{(NXKN}$ which arises in the two dimensional DFT. Specifically, if we consider two vectors U and $W \in \mathrm{R}^{N2}$ defined as $U = \{u_{i,j}\}$ and $W = \{w_{i,j}\}$, for $1 < i, j \leq$ N, then $U = \mathcal{P}W$ implies that $u_{i,j} = w_{j,i}$. Note that $\mathcal{P}^{-1} = \mathcal{P}^T$ since $\mathcal{P}$ is a permutation matrix and therefore is orthogonal.

Theorem *2 The* ***EE*** *decomposition* ***of*** M is given *by*

$$\mathbf{M} = \mathcal{F}\mathcal{P}\mathcal{Q}\Lambda\mathcal{Q}^{-1}\mathcal{P}^T\mathcal{F}^{-1} \tag{3.11}$$

where A ***is*** *the diagonal matrix* ***of eigenvalues of*** M *which, along with the matrix Q, is defined below.*

Proof. By using Eq. (3.9), the matrix M can be written as

$$\mathrm{M} = \text{Tridiag}\,[(1 - 1/2i)I\; F\lambda_i F^{-1}(1 + 1/2i)I] = \mathcal{F}\mathcal{R}\mathcal{F}^{-1} \tag{3.12}$$

where $\mathcal{R}$ is a block tridiagonal matrix given by

$$\mathcal{R} = \text{Tridiag } [(1 - 1/2i)I \; \lambda_i \; (1 + 1/2i)I] . \tag{3.13}$$

Since the block elements of $\mathcal{R}$ are diagonal, it can be transformed to a block diagonal matrix $\mathcal{T}$ using

$$\mathcal{R} = \mathcal{P}\mathcal{P}^T\mathcal{R}\mathcal{P}\mathcal{P}^T = \mathcal{P}(\mathcal{P}^T\mathcal{R}\mathcal{P})\mathcal{P}^T = \mathcal{P}\mathcal{T}\mathcal{P}^T \tag{3.14}$$

where T is defined by

$$T = \text{Diag}\{T_j\} \in \mathbf{R}^{KN\times KN}, \; 1 \leq j \leq N \tag{3.15}$$

$$T_j = \text{Tridiag } [(1 - 1/2i) \; \lambda_{i,j} \; (1 + 1/2i)] \in \mathbf{R}^{K\times K}, \; q \leq i \leq p . \tag{3.16}$$

Let the EE decomposition of T_j be given by

$$T_j = Q_j\Lambda_jQ_j^{-1}. \tag{3.17}$$

Defining $\mathcal{Q} = \text{Diag}\{Q_j\}$ and $\Lambda = \text{Diag}\{\Lambda_j\}$, $1 < j \leq N$, it follows that

$$\mathcal{T} = \mathcal{Q}\Lambda\mathcal{Q}^{-1}. \tag{3.18}$$

Substituting Eq. (3.18) into Eq. (3.14) gives

$$\mathcal{R} = \mathcal{P}\mathcal{T}\mathcal{P}^T = \mathcal{P}\mathcal{Q}\Lambda\mathcal{Q}^{-1}\mathcal{P}^T. \tag{3.19}$$

The EE decomposition of M, given by Eq. (3.,11), follows from substituting Eq. (3.19) into Eq. (3.12). □

The matrix T_j is not symmetric. However, the products of pairs of the corresponding off-diagonal elements are all nonzero and positive. In this sense, T_j is *sign symmetric* and hence has real eigenvalues. Due to this property, *all the eigenvalues and eigenvectors of these matrices can be efficiently computed* by using, for example, the subroutine RT provided by EISPACK [36].

Multiplication of a vector by the matrix of eigenvectors Q_j^{-1} corresponds to the solution of a linear system involving a computational cost of $O(K^3)$. However, following our discussion in Section 2.3, a greater computational efficiency can be achieved by noting that Q_j^{-1}

is the matrix of eigenvectors of matrix T_j^T. Therefore, if the EE decomposition of T_j^T is also computed then the matrix Q_j^{-1} is explicitly obtained and its multiplication by a vector represents a simple matrix-vector multiplication which can be performed with a computational cost of $O(K^2)$. For typical values of K (of the order of 10^2), this scheme results in a two order of magnitude improvement in the computational efficiency. As mentioned before, in a parallel environment and with a sufficient number of processors, the EE decomposition of the matrices T_j^T can be performed in parallel with that of the matrices T_j without increasing the overall computational cost.

3.3 EE Decomposition: Neumann Boundary Condition

We now consider the Neumann boundary condition at the outer domain boundary given by

$$\frac{\partial}{\partial \rho}\psi(p+1,j) = 0. \tag{3.20}$$

Extending the domain by introducing the fictitious points $\psi_{p+2,j}$ and using a centered difference scheme with second-order accuracy, the discretization of the Eq. (3.20) gives

$$\psi_{p+2,j} = \psi_{p,j}. \tag{3.21}$$

Substituting Eq. (3.21) into Eq. (3.2) for $i = p + 1$, it follows that

$$(\Delta\rho)^2\nabla^2\psi(p+1,j) = 2\psi_{p,j} - 2\left[1 + \frac{1}{(i\Delta\phi)^2}\right]\psi_{p+1,j} + \frac{1}{(i\Delta\phi)^2}(\psi_{p+1,j+1} + \psi_{p+1,j-1}). \tag{3.22}$$

The matrix M arising from the discretization of the Laplace operator with the Neumann boundary condition is then a block tridiagonal matrix expressed as

$$\hat{\mathbf{M}} = \begin{bmatrix} B_q & \left[1+\frac{1}{2q}\right]I & & \\ \left[1-\frac{1}{2(q+1)}\right]I & B_{q+1} & \ddots & \\ & \ddots & \ddots & \left[1+\frac{1}{2(q+1)}\right]I \\ & & \left[1-\frac{1}{2p}\right]I & B_p \end{bmatrix} \in \mathbf{R}^{K'N\times K'N} \tag{3.23}$$

where $K' = K + 1$. The similarities between the matrix M and the matrix M imply that the process for computing the EE decomposition for both matrices is similar. To this end, we

define the matrices $\hat{\mathcal{F}}, \hat{\mathcal{F}}^{-1}$, and $\hat{\mathcal{P}}$ similar to the matrices $\mathcal{F}$, $\mathcal{F}^{-1}$, and $\mathcal{P}$ but of dimension $K'N \times K'N$. The matrix M can be written as

$$\hat{\mathbf{M}} = \hat{\mathcal{F}}\hat{\mathcal{R}}\hat{\mathcal{F}}^{-1} \tag{3.24}$$

where $\hat{\mathcal{R}}$ can be reduced to a block diagonal matrix using the transformation

$$\hat{\mathcal{R}} = \hat{\mathcal{P}}(\hat{\mathcal{P}}^T\hat{\mathcal{R}}\hat{\mathcal{P}})\hat{\mathcal{P}}^T = \hat{\mathcal{P}}\hat{\mathcal{T}}\hat{\mathcal{P}}^T. \tag{3.25}$$

$\hat{\mathcal{T}}$ is the block diagonal matrix? = $\text{Diag}\{\hat{T}_j\}, 1 \leq j \leq N$, where

$$\hat{T}_j = \begin{bmatrix} \lambda_{q,j} & 1+\frac{1}{2q} & & \\ 1-\frac{1}{2(q+1)} & \lambda_{q+1,j} & 1+\frac{1}{2(q+1)} & \\ \vdots & \vdots & \vdots & \\ & 1-\frac{1}{2p} & \lambda_{p,j} & 1+\frac{1}{2p} \\ & & 2 & \lambda_{p+1,j} \end{bmatrix} \in \mathbf{R}^{K'\times K'} \tag{3.26}$$

The matrix $\hat{T}_j$ is sign symmetric and, except for the last row, has the same structure as the matrix T'_j. Therefore, the eigenpairs of $\hat{T}_j$ can be efficiently computed. Let the EE decomposition of $\hat{T}_j$ be given by

$$\hat{T}_j = \hat{Q}_j\hat{\Lambda}_j\hat{Q}_j^{-1}. \tag{3.27}$$

Defining $\hat{\mathcal{Q}} = \text{Diag}\{\hat{Q}_j\}$ and $\hat{\Lambda} = \text{Diag}\{\hat{\Lambda}_j\}, 1 \leq j \leq \text{N}$, it follows that

$$\hat{\mathcal{T}} = \hat{\mathcal{Q}}\hat{\Lambda}\hat{\mathcal{Q}}^{-1}. \tag{3.28}$$

Substituting Eqs. (3.25) and (3.28) into Eq. (3.24) gives

$$\hat{\mathbf{M}} = \hat{\mathcal{F}}\hat{\mathcal{P}}\,\hat{\mathcal{Q}}\hat{\Lambda}\hat{\mathcal{Q}}^{-1}\hat{\mathcal{P}}^T\hat{\mathcal{F}}^{-1}. \tag{3.29}$$

As before, it is more efficient to explicitly obtain the matrices $\hat{Q}_j^{-1}$ by computing the EE decomposition of matrices $\hat{T}_j^T$.

3.4 Structure of the Algorithm for the Circular Cylinder

Appendix A provides a pseudo-code listing of the algorithm implementation for the circular cylinder which follows the steps outlined in Figure 1. Additionally, the flow charts in Figures 3 and 4 illustrate how the different steps are computed by taking advantage of the

factored forms of the matrices Θ and S. l?igure 3showsthe determination of Q, Q^{-1}, and S = Diag$\{S_1, S_2, \ldots, S_N\}$ in factored form from the matrices T_j and T_j^T. Figures 4(a),(b), and (c) demonstrate how to use these factored matrices to perform the multiplications in Steps 2, 3, and 4 respectively. These figures also imply that an additional level of parallelism may be exploited in the computation, an issue which is explored more fully in Section 4.2. Note that the SOILR in Figure 4(b) is given by

$$\tilde{\psi}_j^{(m+1)} = 2S_j\tilde{\psi}_j^{(m)} - \tilde{\psi}_j^{(m-1)} + \tilde{f}_j^{(m+1)} \tag{3.30}$$

4 Algorithm Performance for the Circular Cylinder

4.1 Computational Complexity of the Time-Parallel Implementation

In analyzing the performance of the time-parallel algorithm, we assume that M processors are available to fully exploit temporal parallelism. For typical values of M (of the order of 10^3) and K and N (of the order of 102), we have $M >> K$ and $M >> N$. Assuming $M \geq 2N$, the EE decomposition of the 2N sign symmetric tridiagonal matrices T_j and T_j^T, $1 \leq j \leq N$, in Step 1 (Figure 3) can be computed in parallel at a cost of $O(K^2)$, which is the cost required to decompose a single sign symmetric tridiagonal matrix. The computation of the matrices S_j can be also performed in parallel with a cost of O(K). This leads to an overall cost of $O(K^2)$ for Step 1.

The computations in Steps 2 and 4 are fully decoupled in time. Thus, with M processors, they can be computed in parallel for $m = 2$ to M. Referring to Figure 4(a), multiplication by the matrix% involves K one dimensional DFTs, each with a cost of O(N log N), leading to a total cost of $O(KN \log N)$. The N multiplications $\tilde{f}_j^{(m)} = S_j Q_j^{-1} U_{2j}^{(m)}$, $1 \leq j \leq N$, each involves a cost of $O(K^2)$, leading to a total cost of $O(K^2 N)$. Since Step 4 is virtually identical to Step 2 in terms of computational complexity (see Figure 4(c)), the overall cost for each step is $O(KN \log N + K^2 N)$.

The SOILR in Eq. (2.8) involves vectors of dimension KN. With M processors and by using RDA or CRA [30, 31], the cost of parallel solution of Eq. (2.8) is of $O(KN \log M)$. Even assuming very large values for M (of the order of 105), the computational cost of Step

3 is less than that of Steps 2 and 4. Additionally, by exploiting parallelism, the cost of Step 1 is less than that of Steps 2 and 4 (a sequential computation of Step 1 will lead to a cost near] y equal to that of a parallel computation of Steps 2 and 4). Therefore, this represents an example for which the overall computational cost of the time-parallel algorithm is mainly determined by the costs of Steps 2 and 4 which are fully parallelizable in time. In light of these arguments, the computational cost of the time-parallel algorithm while fully exploiting temporal parallelism is given by

$$C_{TP} = a_1 K^2 N + a_2 KN \log N + a_3 KN \log M \qquad (4.1)$$

where a_1, a_2, and a_3 are constants and lower degree terms have been neglected.

4.2 Computational Complexity of the Time- and Space-Parallel Implementation

With the availability of a larger number of processors an additional level of parallelism – spatial parallelism - can also be exploited in the computation. As mentioned in Section 3.4, Figure 4 reveals the structure of the spatial parallelism in the computation.

For a time- and space-parallel computation we consider using ML processors where $L = \mathrm{Max}(N, K)$. With this strategy, the computation of K one dimensional DFTs in Step 2 can now be done in parallel with a cost of a single one dimensional DFT of O(N log N). The remaining N matrix-vector multiplications in Step 2 can be performed in parallel with a cost of a single matrix-vector multiplication of $O(K^2)$. Again, since Step 4 can be performed in a similar fashion, the overall cost of each step is $O(N \log N + K^2)$ for a time- and space-parallel implement ation.

The computation of Eq. (2.8) in Step 3 can be decomposed into a set of decoupled SOILRs, as indicated in Eq. (3.30) and Figure 4(b). Computing these SOILRs in parallel for $j = 1$ to N and exploiting parallelism in each SOILR solution leads to a parallel computational cost of $O(K \log M)$ for this step. Adding the costs of the different steps, the computational cost of the time- and space-parallel implementation of the algorithm, denoted by C_{TSP}, is then

obtained as

$$C_{TSP} = b_1 K^2 + b_2 N \log N + b_3 K \log M \qquad (4.2)$$

where b_1, b_2, and b_3 are constants.

Interestingly, even the space-parallel implementation of Steps 2-4 results in a coarse grain computation with a rather low communication complexity. The space-parallel computation of Step 3 can be performed in a fully decoupled fashion with no communication requirement. In this step, each processor performs the operations for parallel computation of its corresponding SOILR on vectors of dimension K. In Steps 2 and 4, each processor performs an FFT on vectors of dimension N and a multiplication of a K x K matrix by a $K \times 1$ vector.

However, the permutations in Steps 2 and 4 require communication among processors in the space-parallel computation. Consider Step 2 (Figure 4(a)) as an example and let N = K. Before the permutation, each processor (e.g. processor i) computes the vector $U_{1i}^{(m)}$. If these vectors are considered as the columns of a matrix $\mathcal{U}$, then the permutation corresponds to transposing $\mathcal{U}$. In this case, processor i (which initially contained the ith column of matrix $\mathcal{U}$) will receive the ith row of $\mathcal{U}$. The complexity of such a data communication is a function of the processors interconnection structure. With K processors interconnected through a Hypercube topology, the complexity of this matrix transposition is of O(K log K) (see for example [37]). This implies that, with such interconnection topology, even the space-parallel implementation of Steps 2 and 4 remains highly compute bound since its computation complexity of $O(K^2)$ is greater than its communication complexity of $O(K \log K)$.

4.3 Computational Speedup

The speedup of the time-parallel and time- and space-parallel algorithms can be measured with respect to the best sequential explicit method (Yee's algorithm) and the best sequential implementation of the CN method for the problem. The application of Yee's algorithm to the circular cylinder essentially requires a matrix-vector multiplication at each time step. Because the $KN \times KN$ matrix involved in this operation is highly sparse, it can be shown that the computational cost C_{SEY} of the sequential implementation of this algorithm is given

as

$$C_{SEY} = c_1 M'KN \tag{4.3}$$

where c_1 is constant and M' is the number of time steps required to achieve the same level of accuracy as for the CN method. Due to the stability constraints, M' may be much greater than M. The speedup of the time-parallel implementation of our algorithm with respect to the sequential implementation of Yee's algorithm, denoted by SP_1, is then given by

$$SP_1 = \frac{C_{SEY}}{C_{TP}} = \frac{c_1 M'KN}{a_1 K^2 N + a_2 KN \log N + a_3 KN \log M} = \frac{c_1 M'}{a_1 K + a_2 \log N + a_3 \log M}. \tag{4.4}$$

For practical values of K, N, and M, it is generally true that $K > \log N$ and $K > \log M$. Thus, SP_1 can be approximated by

$$SP_1 = \frac{c_1 M'}{a_1 K} = O(M'/K). \tag{4.5}$$

The speedup of the time- and space-parallel implementation of our algorithm with respect to the sequential implementation of Yee's algorithm is given by

$$SP_2 = \frac{C_{SEY}}{C_{TSP}} = \frac{c_1 M'KN}{b_1 K^2 + b_2 N \log N + b_3 K \log M} = \frac{c_1 M'}{b_1 K/N + b_2 (\log N)/K + a_3 (\log M)/N} \tag{4.6}$$

which, again, can be approximately given by

$$SP_2 = \frac{C_{SEY}}{C_{TSP}} = \frac{c_1 M'}{b_1 K/N} = O(M'N/K) \tag{4.7}$$

If $O(K) = O(N)$, which is typically the case, then we have

$$SP_2 = O(M'). \tag{4.8}$$

The sequential solution of Eq. (2.4) at each time step involves two vector additions, each at a cost of $O(KN)$, and one matrix-vector multiplication to form the right-hand side vector. By exploiting the structure of M, the matrix-vector multiplication can also be performed with a cost of $O(KN)$. The solution vector $\psi^{(m+1)}$ is subsequently obtained by solving a linear system. Since the matrices $(I - \alpha \mathbf{M})$ and M have a similar structure, this linear system solution is equivalent to the solution of the Poisson equation in polar coordinates, which can

be performed using the *Fast* Poisson *Solver* [38, 39] with a complexity of $O(KN \log N)$. It then follows that the cost of the best sequential algorithm for solution of Eq. (2.4), denoted as C_{SCN}, is given by

$$C_{SCN} = d_1 MKN \log N + d_2 MKN \tag{4.9}$$

where d_1 and d_2 are constants. The speedup of the time-parallel algorithm with respect to the best sequential algorithm for computation of Eq. (2.4) is then given by

$$SP_3 = \frac{C_{SCN}}{C_{TP}} = \frac{d_1 MKN \log N + d_2 MKN}{a_1 K^2 N + a_2 KN \log N + a_3 KN \log M} = \frac{d_1 M \log N + d_2 M}{a_1 K + a_2 \log N + a_3 \log M} \tag{4.10}$$

which can be approximately given by

$$SP3 = \frac{d_1 M \log N}{a_1 K} = O((M \log N)/K). \tag{4.11}$$

The speedup of the time- and space-parallel algorithm with respect to the best sequential algorithm for computation of Eq. (2.4) is obtained as

$$SP_4 = \frac{C_{SCN}}{C_{TSP}} = \frac{d_1 MKN \log N}{b_1 K^2 + b_2 N \log N + b_3 K \log N} = \frac{d_1 M \log N}{a_1 K/N + a_2 (\log N)/K + a_3 (\log M)/N} \tag{4.12}$$

which can be approximated as

$$SP_4 = \frac{d_1 M \log N}{a_1 K/N} = O((MN \log N)/K). \tag{4.13}$$

Again, assuming N and K to be of the same order, we have

$$SP_4 = O(M \log N). \tag{4.14}$$

Table 1 summarizes the asymptotic computational complexity of the serial and parallel algorithms as well as the number of processors required. As can be seen from Eqs. (4.4)-(4.14), the time-parallel and time- and space-parallel implementation of our algorithm leads to a massive speedup in the computation while resulting in highly coarse grain parallel computation with simple communication and synchronization requirements. For typical values of M' (of the order of 10^4), M (of the order of 10^3), and K and N (of the order of 10^2), the time-parallel implementation of our algorithm leads to more than t wo orders

Table 1: Comparison of serial and parallel algorithms

	Algorithm	Computational Complexity	Number of Processors
Serial	Explicit (Yee's)	$O(M'KN)$	
	Implicit CN	$O(MKN \log N)$	
Parallel	Time-Parallel	$O(K^2N)$	$O(M)$
	Time and Space-Parallel	$O(K^2)$	$O(ML)$

of magnitude speedup in the computation over the best sequential explicit and implicit methods. An even more impressive speedup is possible using the time- and space-parallel implementation of the algorithm which, when used with a massive number of processors, is several orders of magnitude faster than the best sequential algorithm.

It should be mentioned that the performance of the time-parallel algorithm increases for problems demanding a much larger number of time steps. This follows from the fact that time dependence occurs only in the computation of Step 3, with a dependency of $O(\log M)$, while the dominant parts of the computation (Steps 2 and 4) are fully decoupled in time and are therefore independent of M.

5 Generalization of Time-Parallel Approach

Provided that the matrix M is nonsingular, the diagonalization process of Section 2.2 can be applied to the time-stepping procedures for solution of Maxwell's equations. Hence, the main issue in generalization of our approach is not the domain of applicability but rather the computational efficiency. As indicated in Section 2.3, efficient application of the time-parallel approach requires fast schemes for computing the eigenpairs of M and multiplying the matrix of eigenvectors by a vector. In this section, we briefly discuss the implication of these two factors for more general problems.

5.1 Fast Computation of the Eigenpairsof M

Depending on the structure of the spatially determined matrix M, we have identified three main techniques for the efficient computation of its eigenpairs. The first and second techniques are specialized in that they take advantage of the specific structure of M while the third technique is more general since only the sparsity of M is exploited.

a. Analytical Expressions

For a few simple cases involving regular domains, an analytical expression for the eigenpairs of M is known a *priori*. Well-known examples are cases involving two and three dimensional square and cubical domains. Additionally, we have found it possible to develop expressions for some cases which have not been previously explored [28]. Because this approach allows extremely efficient application of the time-parallel approach, it is useful to identify problems which can be solved in this fashion. Nevertheless, it appears that the domain of applicability for this approach remains quite limited.

b. A Divide and Conquer Method

The second technique can be considered as a divide and conquer approach which exploits the specific structure of M. The task is to reduce the computation of eigenpairs of M to the computation of eigenpairs of a set of simpler matrices. Such a divide and conquer approach was applied to the problem in Section 3. As shown in that derivation, this technique is highly efficient with an optimal computational complexity for most cases. Because the EE decomposition for the set of matrices can be performed in parallel, this approach also offers a high degree of coarse grain parallelism. Furthermore, the resulting matrix of eigenvectors can be efficiently multiplied by a vector since it can be obtained as a product of sparse matrices.

c. General Sparse Matrix Techniques

For problems where these methods can not be applied, the highly sparse structure of M must be exploited to efficiently compute its eigenpairs [40]. However, it must be remembered that we are not interested in explicit computation of the matrix of eigenvectors but rather its multiplication by a vector. Therefore, it is more efficient to obtain the matrix of eigenvectors

in a factored form to improve the efficiency of this matrix-vector multiplication. This is a clear departure from most conventional sparse eigenproblem techniques wherein the explicit computation of the matrix of eigenvectors is sought.

5.2 Efficient Matrix-Vector Multiplication

The importance of using efficient techniques for multiplying the matrix of eigenvectors Θ by a vector is perhaps best understood by considering the problem solved in Section 3. For this case, as can be seen from Eq. (3.11), Θ is a full KN x KN matrix, leading a cost of $O(K^2N^2)$ when it is multiplied by a vector. However, by computing Θ in a factored form rather than determining it explicitly, the cost of its multiplication by a vector is reduced to $O(K^2N)$. For a typical value of N (of the order of 10^2), this represents an improvement of two orders of magnitude in computational efficiency, regardless of the degree to which parallelism is exploited in this operation. Such a significant improvement in computational efficiency motivates the identification of those techniques which allow determination of the matrix of eigenvectors in a factored form.

6 Conclusion

In this paper, we have introduced a novel time-parallel approach for solving Maxwell's equations using FDTD techniques. The algorithm provides a massive degree of coarse grain parallelism with simple communication and synchronization requirements, Furthermore, in contrast to previous work which has emphasized the use of explicit FDTD methods, this approach exploits the superior numerical properties of the implicit CN method. The application of the algorithm to solution of a testbed problem has illustrated the massive speedup that can be achieved by exploiting temporal and spatial parallelism. Work is currently underway to implement the algorithm on the Touchstone Delta supercomputer for the circular cylinder problem discussed in this paper. The results of this implementation will be presented in a future correspondence.

In general, however, even the exploitation of full temporal parallelism alone requires a

number of processors which, by far, exceeds that offered by the current generation of massively parallel MIMD architectures. Future generations of these architectures are expected to employ many thousands of processors and to achieve a Teraflop computing capability. Our preliminary results clearly point to a new direction in massively parallel CEM which would enable efficient application of these future architectures to various CEM problems. However, in order to achieve this goal, further research work is needed to extend the domain of efficient applicability of our approach. To this end, the discussion in Section 5 provides a useful framework for further application of the time-parallel computing approach.

ACKNOWLEDGMENT *The research of A. Fijany and J. Barhen was performed at the Center for Space Microelectronics Technology, Jet Propulsion Laboratory, California Institute of Technology, under contract with National Aeronautics and Space Administration (NASA). The support and encouragement of Dr. Paul Messina, Director of the Concurrent Supercomputing Consortium, is greatly acknowledged.*

A Structure of the Algorithm for the Circular Cylinder

Step 1. Compute the EE decomposition of M and Construct S

For j = 1 to N, Do_Parallel

a. Compute the EE decomposition of T_j and $T_j^T \in \mathbf{R}^{K\times K}$

b. Compute $S_j = (I - \alpha\Lambda_j)^{-1}$

End_Do_Parallel

Step 2. Compute Vectors $\tilde{f}^{(m)}$

For m = 2 to M, Do_Parallel

a. Compute $U_1^{(m)} = \mathcal{F}^{-1} f^{(m)}$

For i = q to p, Do_Parallel

$U_{1i}^{(m)} = F^{-1} f_i^{(m)}$

End_Do_Parallel

b, Compute $U_2^{(m)} = \mathcal{P}^T U_1^{(m)}$

c. Compute $\tilde{f}^{(m)} = SQ^{-1}U_2^{(m)}$

For j = 1 to N, Do_Parallel

$\tilde{f}_j^{(m)} = S_j Q_j^{-1} U_{2j}^{(m)}$

End_Do_Parallel

End_Do_Parallel

Step 3. Compute $\tilde{\psi}^{(m)}$

Solve in parallel the SOILR:

$$\tilde{\psi}^{(m+1)} = 2S\tilde{\psi}^{(m)} - \tilde{\psi}^{(m-1)} + \tilde{f}^{(m+1)} \qquad \text{(A1)}$$

This computation can be simplified by solving Eq. (3.30) in Parallel for $j = 1$ to N.

Step 4. Compute $\psi^{(m)}$

For $m = 2$ to M, Do-Parallel

a. Compute $W_1^{(m)} = \mathcal{Q}\tilde{\psi}^{(m)}$

For j = 1 to N, Do_Parallel

$W_{1j}^{(m)} = Q_j\tilde{\psi}_j^{(m)}$

End_Do_Parallel

b. Compute $W_2^{(m)} = \mathcal{P}W_1^{(m)}$

c. Compute $\psi^{(m)} = \mathcal{F}W_2^{(m)}$

For i = q to p, Do-Parallel

$\psi_i^{(m)} = FW_{2i}^{(m)}$

End_Do_Parallel

End_Do_Parallel

References

[1] E. K. Miller. Solving bigger problems by decreasing the operation count and increasing the computation bandwidth. *Proc. IEEE, Special Issue on Electromagnetic*, 79(10):1493–1504, Ott 1991.

[2] K. S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Trans. Antennas Propag., AP-14:302-307,* May 1966.

[3] A. Taflove and M. E. Brodwin. Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations. *IEEE Trans. Microwave Theory Tech., MTT-23:623-630,* Aug 1975.

[4] K. Umashankar and A. Taflove. A novel method to analyze electromagnetic scattering of complex objects. *IEEE Trans. Electromagnetic Compat.*, EMC-24:397–405, *Nov 1982.*

[5] A. H. Mohammadian, V. Shankar, and W. F. Hall. Computation of electromagnetic scattering and radiation using a time-domain finite-volume discretization procedure. *Comput. Physics Commun.*, 68:175–196, Ott 1991.

[6] V. Shankar, A. H. Mohammadian, W. F. Hall, and R. Erickson. CFD spinoff Computational electromagnetic for radar cross section (RCS) studies. In *Pmt. AIAA Applied Aerodynamics Conf., 1990.*

[7] R. H. Calalo et al. Hypercube matrix computation task, Aug 1988.

[8] J. E. Patterson et al. Concurrent electromagnetic scattering analysis. In *Proc. AIAA Computers in Aerospace VII Conf.*, Ott 1989.

[9] W. J. Buchanan, N. K. Gupta, and J. M. Arnold. Simulation of radiation from a microstrip antenna using three dimensional finite-difference time-domain (FDTD) method. In *Proc. IEE 8th Int. Conf. on Antennas and Propagation,* pages 639-642, Heriot-Watt Univ., UK, 1993.

[10] A. Taflove and K. R. Umashankar. Review of FD-TD numerical modeling of electromagnetic wave scattering and radar cross section. *Proc. of IEEE, Special Issue on Radar Cross Section of Complex Object,* 77(5):682–699, May 1989.

[11] H. Vinh, H. A. Dwyer, and C. P. Van Dam. Finite-difference algorithms for time-domain Maxwell's equations-A numerical approach to RCS analysis. In *Proc. 23rd AIAA Plasmadynamics & Laser Conf.*, pages 1-8, Jul 1992,

[12-J R. L. Burden and J. D. Faires. *Numerical Analysis.* PWS-KENT Publishing Co., Boston, MA., 1989.

[13] J. M. Ortega and R. G. Voigt. *Solution of Partial Differential Equations on Vector and Parallel Computers,* SIAM, 1984.

[14] E. Gallopoulos and Y. Saad. On the parallel solution of parabolic equations. In *Proc. ACM Int. Conf. on Supercomputing,* pages 17--28, Jun 1989.

[15] S. Vandewalle, R. Van Driessche, and R. Piessens. The parallel performance of standard parabolic marching schemes. *Int. J. High Speed Computing,* 3(1):1–29, *1991.*

[16] G. Rodrigue. A parallel first-order method for parabolic partial differential equations. In J. S. Kowalik, editor, *High-Speed Computation,* pages 329-342. Springer-verlag, 1984.

[17] D. J. Evans. Alternating group explicit methods for the diffusion equation. *Appl. Math. Modeling,* 9:201–206, *1985.*

[18] S. M. Serbin. A scheme for parallelizin g certain algorithms for the linear inhomogeneous heat equation. *SIAM J. Sci. Stat. Comput.*, 13(2) :449–458, Mar 1992.

[19] E. Lelarasmee, A. Ruheli, and A. L. Sangiovanni-Vincentelli. The waveform relaxation method for the time domain analysis of large scale integrated circuits. *IEEE Trans. Computer-Aided Design,* 1:131–145, *1982.*

[20] J. H. Saltz and V. K. Nail. Towards developing robust algorithms for solving partial differential equations on MIMD machines. *Parallel Computing,* 6:19–44, *1988.*

[21] D. E. Womble. A time-stepping algorithm for parallel computers. *SIAM J. Sci. Stat. Comput.*, 11(5):824–837, *1990.*

[22] W. Hackbusch. Fast numerical solution of time-periodic parabolic problems by a multigrid method. *SIAM J. Sci. Stat. Comput.*, 2:198–206, *1981.*

[23] *G.* Horton and R. Knirsch. A time-parallel multigrid-extrapolation method for parabolic partial differential equations. *Parallel Computing,* 18:21–29, *1992.*

[24] *S.* Vandewalle and R. Piessens. Efficient parallel algorithms for solving initial-boundary value and time-periodic parabolic differential equations. *SIAM J. Sci. Stat. Comput.*, 13(6):1330–1346, Nov 1992.

[25] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[26] A. Fijany. Time-parallel algorithms for solution of linear parabolic PDEs. In *Proc. Int. Conf. Parallel Processing (ICPP),* volume III, pages 51-55, Aug 1993.

[27] A. Fijany, J. Barhen, and N. Toomarian. Massively parallel algorithms for solution of the Schrodinger Equation. In *Proc. of Int. Parallel Processing Symp. (IPPS),* Cancun, Mexico, Apr 1994.

[28] A. Fijany, J. Barhen, and N. Toomarian. On the structure of time-parallel algorithms for solution of evolutionary partial differential equations, In preparation.

[29] N. Toomarian, A. Fijany, and J. Barhen. Time parallel solution of linear partial differential equations on the Intel Touchstone Delta supercomputer. *to appear in J. of Concurrency, 1994.*

[30] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. on Computers,* C-22(8):786–793, *1973.*

[31] R. Hockney and C. Jesshope. *Parallel Computers.* Adam Hilger Ltd., 1981.

[32] G. Mur. Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic field equations. *IEEE Trans. Electromagnetic Compat., EMC-23:377-382,* Nov *1981.*

[33] D. Givoli. Non-reflecting boundary conditions. *J, Comput. Physics,* 94:1–29, May 1991.

[34] X. Zhang, J. Fang, K. K. Mei, and Y. Liu. Calculations of the dispersive characteristics of microstrips by the time-domain finite difference method. *IEEE Trans. Microwave Theory Tech.,* 36(2):263–267, Feb 1988.

[35] C. Van Loan. *Computational frameworks for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[36] B. T. Smith et al. *Matrix Eigensystem Routines-Eispack Guide, 2nd Edition.* Springer Verlag, 1976.

[37] 0. A. McBryan and E. F. Van De Velde. Hypercube algorithms and implementation. *SIAM J.* Sci. *Stat. Comput.*, 8(2):227–287, Mar 1987.

[38] P. N. Swarztrauber and R. A, Sweet. The direct solution of discrete Poisson equation on a disk, *SIAM J. Numer. Anal.*, 10(5):900–907, Ott 1973.

[39] B. Buzbee, G. Golub, and C. Nielson. On direct methods for solving Poisson equations. *SIAM J. Numer. Anal.,* 7:627–656, *1970.*

[40] *S.* Pissanetzky. *Sparse Matrix Technology.* Academic Press, 1984.

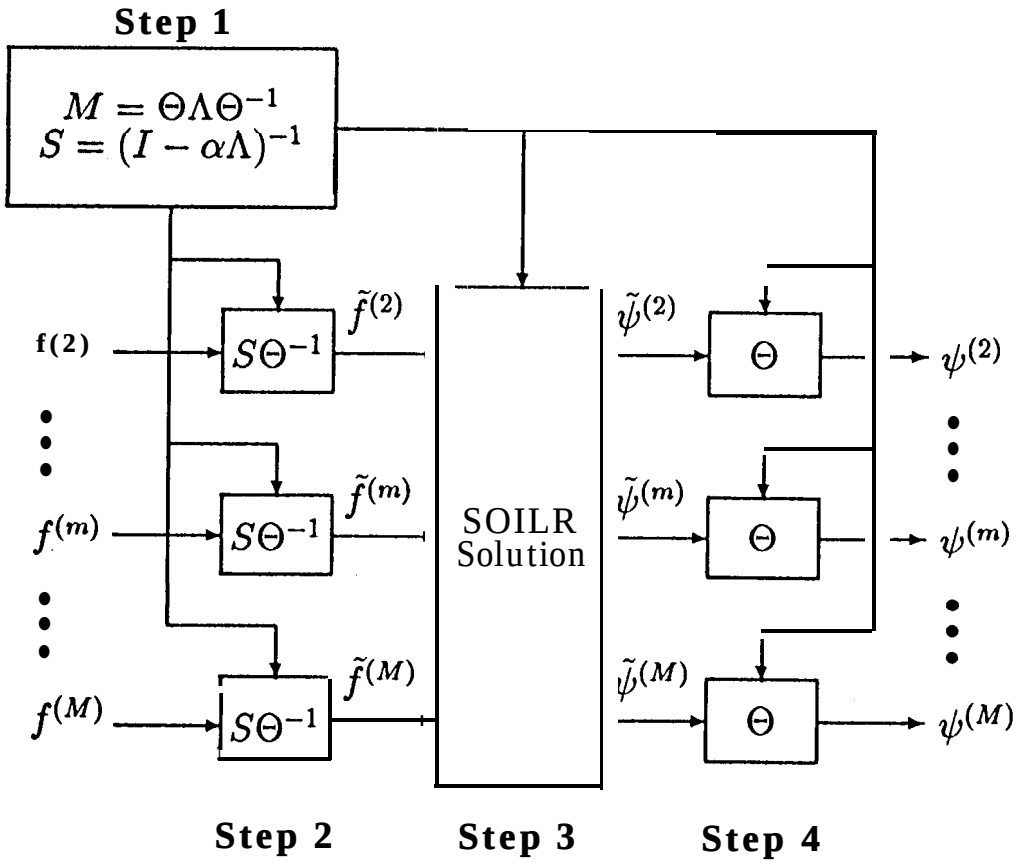


Figure 1: Flow diagram illustrating the computational structure of the time-parallel algorithm.

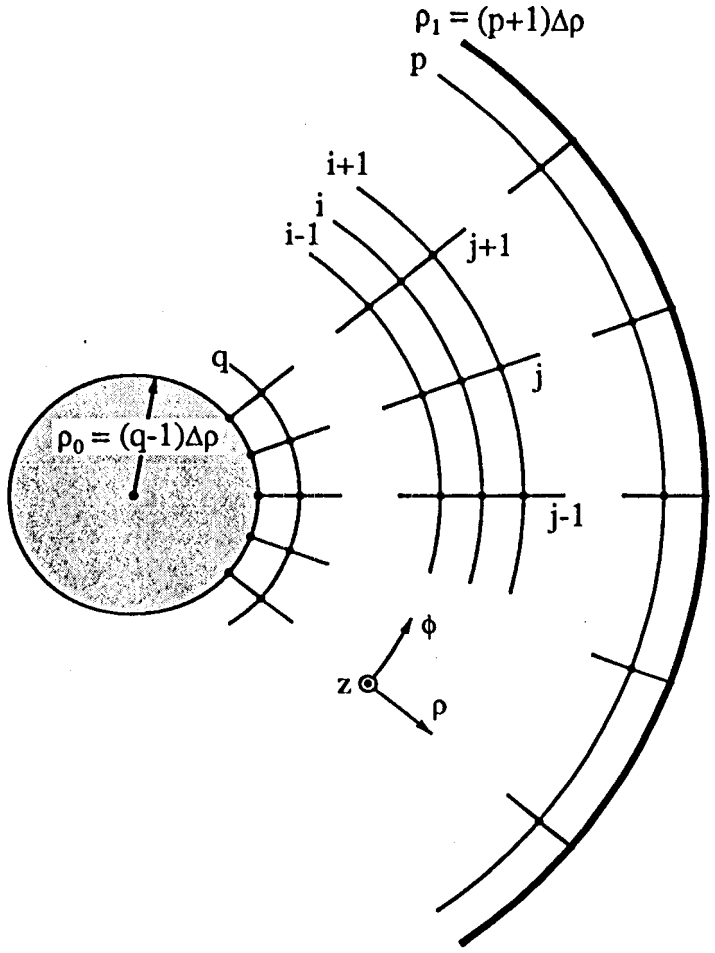


Figure 2: Geometry and gridding for the CN solution of Maxwell's equations near a perfectly conducting circular cylinder.

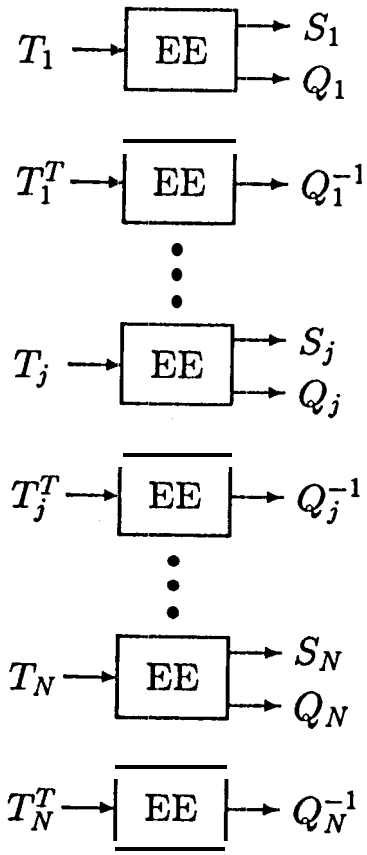
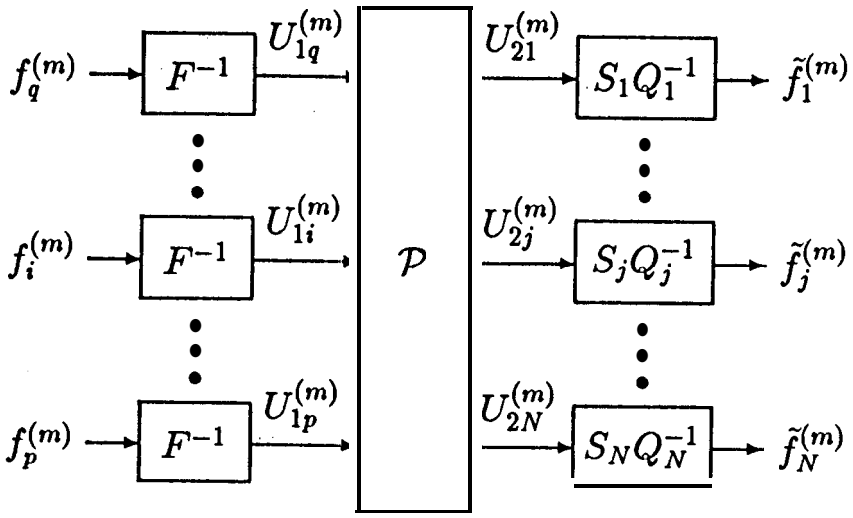
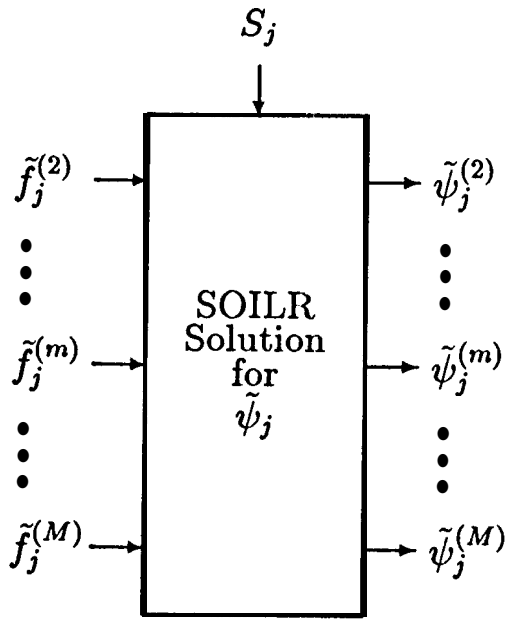


Figure 3: Flow chart illustrating the parallel computation of Step 1 by obtaining the matrices Q, Q^{-1}, and S in factored form.


$f_q^{(m)}$
F^{-1}
$U_{1q}^{(m)}$
$f_i^{(m)}$
F^{-1}
$U_{1i}^{(m)}$
$f_p^{(m)}$
F^{-1}
$U_{1p}^{(m)}$
$\mathcal{P}$
$U_{21}^{(m)}$
$S_1Q_1^{-1}$
$\tilde{f}_1^{(m)}$
$U_{2j}^{(m)}$
$S_jQ_j^{-1}$
$\tilde{f}_j^{(m)}$
$U_{2N}^{(m)}$
$S_NQ_N^{-1}$
$\tilde{f}_N^{(m)}$

(a)


S_j
$\tilde{f}_j^{(2)}$
$\tilde{f}_j^{(m)}$
$\tilde{f}_j^{(M)}$
SOILR
Solution
for
$\tilde{\psi}_j$
$\tilde{\psi}_j^{(2)}$
$\tilde{\psi}_j^{(m)}$
$\tilde{\psi}_j^{(M)}$

(b)

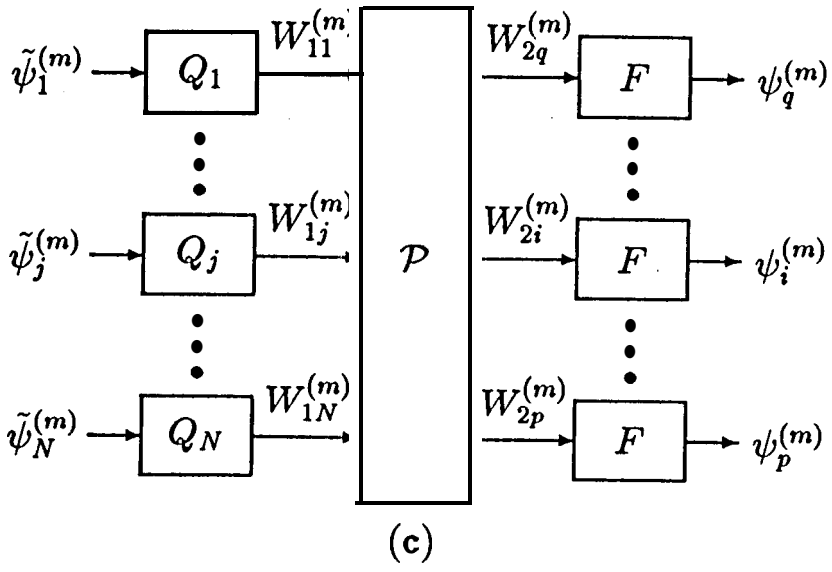


Figure 4: Flow diagram illustrating the space-parallelism in the computation of (a) Step 2, (b) Step 3, and (c) Step *4*.